UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/858,240	05/15/2001	Philip J. Goward	WBG01-0004	1724

22835 7590 07/19/2004

PARK, VAUGHAN & FLEMING LLP
508 SECOND STREET
SUITE 201
DAVIS, CA 95616

EXAMINER	
MOIDUDDIN, NOREEN	

ART UNIT	PAPER NUMBER
2124	6

DATE MAILED: 07/19/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/858,240	GOWARD ET AL.	
	Examiner	Art Unit	
	Noreen Moiduddin	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *15 May 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-48* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) _____ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-48 are pending and have been examined. The priority date considered for the application is 15 March 2001.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

3. Claims 1, 3-9, and 12-14 provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 1, 4-13 of copending Application No. 09/858249 (hereinafter '249 application). Although the conflicting claims are not identical, they are not patentably distinct from each other because they are directed to substantially the same invention and recite only obvious differences, which would have been obvious to one of ordinary skill in the art of program development at the time of the invention.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

The corresponding claims are as follows:

Instant claim in Pending Application	Claim in '249 Application
1, 17, 33	13
3, 19, 35	4
4, 20, 36	5
5, 21, 37	6
6, 22, 38	7
7, 23, 39	8
8, 24, 40	9
9, 25, 41	10
12, 28, 44	11
13, 29, 45	12
14, 30, 46	1

Per claim 1:

Claim 13 in '249 application recites steps of receiving, the distributed application ("receiving the distributed application" in step (a) of '249 application claim 1), automatically identifying, ... any distributed components, ... ("identifying any distributed components" in preamble of '249 application claim 13); ..., automatically deploying the distributed component, ... ("... deploying the distributed component," in claim 13 of '249 application), identifying the remote location, ... ("identifying a remote location, ..." in claim 13 of '249 application), and causing the distributed component to be deployed to the remote location ("and causing the distributed component to be deployed to the remote location." In claim 13 of '249 application); whereby a programmer of the distributed application does not have to enter explicit commands to deploy distributed components to remote locations ("whereby a developer of the distributed application does not explicitly communicate references between the distributed components in order to interlink the distributed components." In claim 1 of '249 application) in this instant claim.

Instant claim 1 does not recite the steps of automatically determining, ... a set of dependencies, ..., wherein a dependency between a first distributed component and a second distributed component indicates that the first distributed component refers to the second distributed component; and automatically ensuring, ... a remote distributed component, ... located on another computer system has a reference to the remote distributed component in claim 13 of '249 application. It would have been obvious for one of ordinary skill in the art of program development at the time the instant invention

was made to modify the '249 application's method by omitting these steps along with their functions.

Per claim 3:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, wherein identifying distributed components that need to be deployed to remote locations involves examining a deployment specifier that indicates where each of the distributed components that make up the distributed application is to be deployed ("wherein determining the set of dependencies involves examining a deployment specifier that indicates where each of the distributed components that make up the distributed application is to be deployed." claim 4 of '249 application).

Per claim 4:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, wherein the distributed application is specified in terms of a component-behavior model; ("wherein the distributed application is specified in terms of a component-behavior model;" claim 5 of '249 application) wherein the component-behavior model specifies components, which are separately deployable pieces of software that can be used to make up an application; ("wherein the component-behavior model specifies components, which are separately deployable pieces of software that can be used to make up an application;" claim 5 of '249 application) and wherein the component-behavior model also specifies behaviors that define a response to an event,

wherein the response can include activating a component ("and wherein the component-behavior model also specifies behaviors that define a response to an event, wherein the response can include activating a component." Claim 5 of '249 application).

Per claim 5:

The rejection of claim 4 is incorporated, and further, this claim recites the additional limitation, wherein activating the component involves invoking a method defined by the component ("The method of claim 5, wherein activating the component involves invoking a method defined by the component" claim 6 of '249 application).

Per claim 6:

The rejection of claim 4 is incorporated, and further, this claim recites the additional limitation, wherein an event can be generated by a component or a behavior ("The method of claim 5, wherein an event can be generated by a component or a behavior." claim 7 of '249 application).

Per claim 7:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, wherein receiving the distributed application involves receiving the distributed application after the distributed application has been modified during a development process ("wherein receiving the distributed application involves receiving the distributed application after the distributed application has been modified during a development

process;" claim 8 of '249 application); and wherein identifying distributed components that need to be deployed to remote locations involves, determining if any distributed components have been modified, and then determining where distributed components that have been modified are to be deployed ("and wherein determining the set of dependencies involves, determining changes to the set of dependencies caused by modifications to the distributed application;" claim 8 of '249 application); whereby only distributed components that have been modified during the development process are deployed ("whereby only dependencies that have been changed during the development process cause new references to be communicated." claim 8 of '249 application).

Per claim 8:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, wherein receiving the distributed application involves: authoring the distributed components that make up the distributed application ("wherein receiving the distributed application involves: authoring the distributed components that make up the distributed application;" claim 9 of '249 application); and creating a deployment specifier that indicates where each of the distributed components is to be deployed ("and creating a deployment specifier that indicates where each of the distributed components is to be deployed." claim 9 of '249 application).

Per claim 12:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, wherein each of the distributed components can include one of: an Enterprise JavaBean (EJB); a Distributed Component Object Model (DCOM) object; and a Common Object Request Broker Architecture (CORBA) object ("The method of claim 1, wherein each of the distributed components can include one of: an Enterprise JavaBean (EJB); a Distributed Component Object Model (DCOM) object; and a Common Object Request Broker Architecture (CORBA) object." claim 11 of '249 application).

Per claim 13:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, wherein a distributed programming component is an Enterprise JavaBean (EJB) that is encapsulated as a JavaBean by combining functionality of a home interface and a remote interface of the EJB into the JavaBean ("The method of claim 1, wherein a distributed programming component is an Enterprise JavaBean (EJB) that is encapsulated as a JavaBean by combining functionality of a home interface and a remote interface of the EJB into the JavaBean." claim 12 of '249 application).

Per claim 14:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, further comprising: determining a set of dependencies between distributed

components that make up the distributed application ("determining under computer control a set of dependencies between
distributed components that make up the distributed application," claim 1 of '249 application), wherein a dependency between a first distributed component and a second distributed component indicates that the first distributed component refers to the second distributed component ("wherein a dependency between a first distributed component and a second distributed component indicates that the first distributed component refers to the second distributed component;" claim 1 of '249 application) and ensuring that each distributed component that depends on a remote distributed component located on another computer system has a reference to the remote distributed component ("ensuring under computer control that each distributed component that depends on a remote distributed component located on another computer system has a reference to the remote distributed component;" claim 1 of '249 application).

Per claim 9:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, wherein receiving the distributed application involves receiving the distributed application during execution of the distributed application ("wherein receiving the distributed application involves receiving the distributed application during execution of the distributed application." claim 10 of '249 application), wherein the distributed components that make up the distributed application are not necessarily deployed prior to executing the distributed application.

As per claims 17, 19-25, and 28-30, there are computer-readable storage medium claims corresponding to the method recited in 1, 3-9, and 12-14 respectively and they are rejected under the same rationale.

As per claims 33, 35-41, and 44-46, there are apparatus claims corresponding to method claim 1, 3-9, 12-14 respectively and they are rejected under the same rationale.

4. Claims 2, 18, and 34 provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1, and 12 of copending Application No. 09/858249 in view of Guthrie et al (U.S. 6,385,661). This is a provisional obviousness-type double patenting rejection.

Per claim 2:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, further comprising: automatically determining if any distributed components of the distributed application have not been encapsulated as local components; and for each distributed component that has not been encapsulated as a local component, automatically encapsulating the distributed component as a local component, so that the distributed component appears to be a local component.

Claim 12 of application '249 recites an example of automatically encapsulating the distributed component as a local component, so that the distributed component appears to be a local component.

Claim 12 of application '249 does not recite the claim limitation wherein, for each distributed component that has not been encapsulated as a local component, automatically encapsulating the distributed component as a local component.

Guthrie teaches an analogous method of enabling local components to implement remote components (column 3, lines 44-46, column 5, lines 23-29, column 1, lines 65-67) on a distributed object oriented system the step wherein each distributed component that has not been encapsulated as a local component, automatically encapsulating the distributed component as a local component ("if remote proxy class does not exist on client system, ...remote proxy class 23 is generated" refer to figure 2 of Guthrie, item 32 and column 5, lines 46-50, a remote proxy class encapsulates a component as a local component).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to determine if any distributed components of the distributed application have not been encapsulated as local components and automatically encapsulate those distributed components, to allow a client system to disregard the location of distributed components of a requested object and the communication details (Guthrie column 1, lines 65-67).

As per claim 18, there is a computer-readable storage medium claim corresponding to the method claim in 2 and it is rejected under the same rationale.

As per claim 34 there is an apparatus claim corresponding to method claim 2 and it is rejected under the same rationale.

5. Claims 10, 11, 26, 27, 42, and 43 provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1, and 8 of copending Application No. 09/858249 in view of Shah et al. (U.S. 6,269,396).

This is a provisional obviousness-type double patenting rejection.

Per claim 11:

The rejection of claim 1 is incorporated, and further, this claim recites the additional limitation, wherein causing the distributed component to be deployed to the remote location involves communicating with an application server at the remote location through a deployment server at the remote location, wherein the deployment server operates by: halting an application server process; loading files for the distributed component onto the application server; setting preferences on the application server for the distributed component; and restarting the application server process.

Claim 8 of Application '249 does not recite the instant claim limitation of claim 11, wherein causing the distributed component to be deployed to the remote location

involves communicating with an application server at the remote location through a deployment server at the remote location, wherein the deployment server operates by: halting an application server process; loading files for the distributed component onto the application server; setting preferences on the application server for the distributed component; and restarting the application server process.

Shah et al teaches an analogous system in a distributed network, wherein steps of deploying (column 4, lines 66-67, column 5, line 1, and column 6, lines 47-49) a distributed component (configurable element set, column 5, line 2, column 6, lines 49-51) to a remote location (one or more nodes, column 5, line 1, figure 24, item 442, and column 6, lines 33-35, lines 45-47) involves communicating with an application server (network figure 1, item 16) at the remote location through a deployment server (telecom platform, figure 1, items 14 and 15, telecom platform manager column 6, lines 45-46) at the remote location, wherein the deployment server (telecom platform, telecom platform manager) operates (manages coloumn5, lines 22-24) by: halting an application server process loading files for the distributed component onto the application server ("HALTED state" (column 5, lines 53)); setting preferences on the application server for the distributed component (configuration of nodes, column 7, lines 25-27, and lines 39-41); and restarting the application server process (a shutdown node (Halted) may recover automatically, column 5, lines 53-54).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to apply the method of halting an application server process, setting preferences (configuration of nodes), and restarting the application server

process recited by Shah, to distribute an distributed application during development as recited by claim 8 of '249 application.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to use the method and system disclosed by Shah to allow developers to write applications without having to master the intricacies of the underlying services (Shah column 4, lines 13-17).

Per claim 10:

The rejection of claim 11 is incorporated, and further, this claim recites the additional limitation, wherein causing the distributed component to be deployed to the remote location involves communicating with an application server at the remote location through an administration protocol. The limitations of claim 10 have already been addressed in connection with instant claim 11. Instant claim 11 of pending application recites an example of an administration protocol used to communicate with an application server, when a distributed component is deployed.

As per claims 26, and 27 there are computer-readable storage medium claims corresponding to the method claim of 10, and 11, respectively and are rejected under the same rationale.

As per claims 42, and 43 there are apparatus claims corresponding to the method claim claims 10, and 11, respectively and are rejected under rejected under the same rationale.

6. Claims 15-16, 31-32, and 47-48 provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 5 of copending Application No. 09/858249 in view of Kon et al ("Supporting Automatic Configuration of component-Based Distributed Systems" 1999) (hereinafter: Kon). This is a provisional obviousness-type double patenting rejection.

Per claim 15:

Claim 5 of application '249 recites the limitation of instant claim 15 wherein the component-behavior model specifies components, ...("wherein the component-behavior model specifies components, ..." claim 5 of '249 application); wherein the component-behavior model also specifies behaviors that activate components in response to events ("wherein the component-behavior model also specifies behaviors that define a response to an event, wherein the response can include activating a component." claim 5 of '249 application).

Claim 5 of application '249 does not recite a method for deploying a component-behavior model within a distributed computer system, comprising: receiving a specification for the component-behavior model, events generated by components or behaviors, and identifying components within the component-behavior model to be

deployed to remote locations; and for each component to be deployed to a remote location, identifying the remote location, and causing the component to be deployed to the remote location.

Kon teaches an analogous method for deploying a component-behavior model within a distributed computer system, comprising: receiving a specification for the component-behavior model ("deployment of the ComponentConfigurator framework (model) in a reflective ORB (distributed computer system)", Kon, Part 3.0), events generated by components or behaviors (figure 2, eventOnHookedComponent() and eventOnClient()), and identifying components within the component-behavior model to be deployed to remote locations (see Part 2.2 of Kon; "a component configurator may be able to refer to components running on, ... different machines (remote locations) in a distributed system); and for each component to be deployed to a remote location, identifying the remote location, and causing the component to be deployed to the remote location ("A distributed resource manager uses the specifications (deployment specifier) of the component hardware requirements to decide in which machine (identifying the remote location) the component (components within the component-behavior model) should be loaded (deployed)" (Kon, Part 3.2)).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to deploy the component-behavior model recited by claim 5 of '249 application, by receiving a specification for the component-behavior model, wherein the component-behavior model specifies behaviors that activate components in response to events, recited by claim 5 of '249 application, where the events are generated by

components or behaviors disclosed by Kon, and the model identifies components within the component-behavior model to be deployed to remote locations. The deployed model as specified by Kon would identify a remote location for a component and cause deployment of the component to the remote location.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to use the ComponentConfigurator framework disclosed by Kon to automate the installation and configuration of new components (Kon Part 2).

Per claim 16:

The rejection of claim 15 is incorporated, and further, this claim recites the additional limitation, wherein identifying components to be deployed to remote locations involves examining a deployment specifier that indicates where each of the components within the component-behavior model is to be deployed ("A distributed resource manager uses the specifications (deployment specifier) of the component hardware requirements to decide in which machine (where) the component (components within the component-behavior model) should be loaded (deployed)" (Kon, Part 3.2)).

As per claims 31, and 32 there are computer-readable storage medium claims corresponding to method claim 15 and 16, respectively and are rejected under the same rationale.

As per claims 47, and 48 there are apparatus claims corresponding to method claim 15, and 16, respectively and are rejected under the same rationale.

Claim Rejections - 35 USC § 102

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 1, 3-10, 12, 14, 17, 19-26, 28, 30, 33, 35-42, 44, 46 are rejected under 35 U.S.C. 102(b) as being anticipated by Fowlow et al (U.S. 5,991,535).

As per claim 1, Fowlow teaches a method comprising:

(a) receiving the distributed application (constructing object oriented application software, column 3, lines 24-25).

(b) automatically ("the computer code effective to implement the relationship (relationship between a selected component and a pre-existing component in a distributed system) is generated (automatically) when the application program is run" column 3, lines 39-41) **identifying under computer control any distributed components within the distributed application that need to be deployed to remote locations** ("A method, apparatus, and program code (under computer control) visually constructs object-oriented application (distributed application) software to be installed

(deployed) on a distributed (remote locations) object system," Abstract lines 1-3, and column 3, lines 23-25).

(c) for each distributed component that needs to be deployed to a remote location, automatically deploying the distributed component under computer control by,

- **identifying the remote location for the distributed component** ("ORB provides (identifies) location (remote location) and transport mechanisms and facilities necessary to deliver a call from a client to a servant (target object (distributed object)) " column 6, lines 18-20).
- **causing the distributed component to be deployed to the remote location** ("Fig. 9 is an illustration of a method for composing object-oriented applications for deployment on a distributed object system in accordance with the present invention" column 5, lines 15-17), whereby a programmer (the programmer, column 11, line 41) of the distributed application (object oriented application, column 3, lines 24-25) does not have to enter explicit commands to deploy distributed components to remote locations ("relieving the programmer of laborious task of locating the appropriate objects across the distributed object system" (column 11, lines 43-46)).

As per claim 3, the rejection of claim 1 is incorporated and further, Fowlow in reference to McDonald et al (U.S. 5,915,113), incorporated herein by reference for all purposes (Fowlow column 6, lines 40-43) teaches **the method of claim 1, wherein identifying distributed components that need to be deployed to remote locations involves**

examining a deployment specifier that indicates where each of the distributed components that make up the distributed application is to be deployed ("Based on a visual display of an application that shows program objects and the connections or interactions between the objects, an internal representation (deployment specifier) of the application is defined (abstract McDonald). Scanning the internal representation (deployment specifier) of the visually partitioned program, distributed connections (distributed components that need to be deployed to remote locations) are located. For each distributed connection (distributed component that need to be deployed to a remote location) in the internal representation (deployment specifier), the server stub representation is located (indicates where a distributed component needs to be deployed) (McDonald column 7, lines 15-20). Deploying in this context refers to updating "pre-existing" distributed components).

As per claim 4, the rejection of claim 1 is incorporated and further, Fowlow teaches **the method of claim 1,**

- **wherein the distributed application is specified in terms of a component-behavior model** ("The composition builder is coupled with a component service that provides the user or programmer access to objects available on the distributed object system" The catalog is an inventory of the software resources available to the programmer on the distributed object system (column 9, lines 39-47)).
- **wherein the component behavior model** (component service (column 9, lines 40)) **specifies components** ("objects available on the distributed object system"

(column 9, lines 40)), **which are separately deployable pieces of software that can be used to make up an application;**

- **wherein the component behavior model also specifies behaviors that define a response to an event, wherein the response can include activating a component** ("The catalogue provides information to the programmer regarding the function and implementation (behavior) of the objects referenced by the components contained in the catalogue" (column 9, lines 47-50)).

As per claim 5, the rejection of claim 4 is incorporated and further, Fowlow teaches **the method of claim 4, wherein activating the component involves invoking a method defined by the component** ("A developer uses an interface definition language to define an interface for an ORB object, provides a developer object implementation that implements that object's behavior, and then uses the object development facility in order to produce an ORB object implementation. At run time, an instance of this ORB object is created that will utilize (invoke) this ORB object implementation (method defined by component)" (column 6, lines 48-54)).

As per claim 6, the rejection of claim 4 is incorporated and further , Fowlow teaches **the method of claim 4, wherein an event can be generated by a component or a behavior** (A developer uses an interface definition language to define an interface for an ORB object, provides a developer object implementation that implements that object's behavior (column 6, lines 48-50). "As will be known to those of skill in the

object programming arts, objects communicate amongst themselves by passing and operating upon object references (components)"(column 11, lines 11-14)).

As per claim 7, the rejection of claim 1 is incorporated and further, Fowlow teaches **the method of claim 1, wherein receiving the distributed application involves receiving the distributed application after the distributed application has been modified** (modified if necessary, column 11, line 60) during a development process (constructing object oriented application software, column 3, lines 24-25); **and wherein identifying distributed components that need to be deployed to remote locations involves, determining if any distributed components have been modified** (identify the relationship among the objects comprise the current composition, column 11, lines 56-60) **and then determining where distributed components that have been modified are to be deployed** (determining the appropriate arguments in syntax necessary to establish communications among those objects, column 11, lines 44-46) **whereby only distributed components that have been modified during the development process are deployed** (modified if necessary, column 11, line 60).

As per claim 9, the rejection of claim 1 is incorporated and further, Fowlow teaches **the method of claim 1, wherein receiving the distributed application involves receiving the distributed application during execution of the distributed application** (figure 1, item 26), **wherein the distributed components that make up the distributed application are not necessarily deployed prior to executing the**

distributed application (the relationship is generated (deployed) when the application program is run, lines 46-47 column 3).

As per claim 8, the rejection of claim 1 is incorporated and further, Fowlow teaches:

(a) authoring the distributed components that make up the distributed application ("the present invention provides a computer-implemented method for constructing (authoring) object-oriented application software (distributed application) to be installed on a distributed object system" column 3 of Fowlow, lines 16-18)

(b) and creating a deployment specifier that indicates where each of the distributed components is to be deployed (A composition worksheet (deployment specifier) containing parts ("placeholder" for an instance of the object type referencing a distributed component) is used to facilitate the design of applications as well as their implementation (deployment of distributed components) (column 11, lines 60-63). The parts provide a reference (indicate where) to each of the components (distributed components) obtained from a component catalog (figure 5, item 550)).

As per claim 10, the rejection of claim 1 is incorporated and further, Fowlow teaches **the method of claim 1, wherein causing the distributed component to be deployed to the remote location involves communicating with an application server at the remote location through an administration protocol** (internet inter-ORB protocol column 7, lines 54-56, ORB column 6, lines 18-21).

As per claim 12, the rejection of claim 1 is incorporated and further, Fowlow teaches **each of the distributed components can include a Common Object Request Broker Architecture (CORBA) object** (ORG, line 22 column 2, ORB objects, line 15 column 8; CORBA specifications, line 5 column 10).

As per claim 14, the rejection of claim 1 is incorporated and further, Fowlow teaches **the method of claim 1, further comprising: determining a set of dependencies** (links) **between distributed components** (parts) **that make up the distributed application** (object oriented application software),
wherein a dependency between a first distributed component and a second distributed component indicates that the first distributed component refers to the second distributed component; ("the step of linking (dependency) compromises defining a connection between a plug on a first part (first distributed component) and a socket on a second part (second distributed component) column 3, lines 51-53. The connection between the plug and socket shows the first distributed component refers to the second distributed component.)
and ensuring that each distributed component (part) **that depends on a remote distributed component** (pre-existing objects provided on a distributed object system) **located on another computer system** (distributed object system) **has a reference to the remote distributed component** (pre-existing object). Refer to figure 5, items 550, 552, and '552. Item 550 represents a component catalog. The catalog facility contains components that provide reference to pre-existing objects (remote distributed

components) provided on a distributed system (column 3, lines 22-26). Item 552 is a component that provides a reference to a pre-existing object and 552' is a new part derived from a component (column 11, lines 30-35). By definition a part (distributed component) is a "placeholder" for an instance of the object type which the part and component both reference (column 11, lines 34-35).

As per claims 17, 19-26, 28, 30, they are computer-readable storage medium corresponding to method claim 1, 3-10, 12, and 14 respectively and further, these claims are rejected by the same rationale.

As per claims 33, 35-42, 44, 46, they are apparatus claims corresponding to method claim 1, 3-10, 12, and 14, respectively and further, these claims are rejected by the same rationale.

Claim Rejections - 35 USC § 102

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 15, 16, 31, 32, 47, and 48 rejected under 35 U.S.C. 102(b) as being anticipated by Kon et al ("Supporting Automatic Configuration of component-Based Distributed Systems" 1999) (hereinafter: Kon).

As per claim 15, Kon and Campbell teach a method for:

receiving a specification for the component behavior model ("deployment of the ComponentConfigurator framework (model) in a reflective ORB (distributed computer

system)", Kon Part 3.0) **wherein the component behavior model specifies components** (figure 2) **which are separately deployable pieces of software that can be used to make up an application; wherein the component behavior model also specifies behaviors that activate components in response to events generated by components or behaviors** (figure 2, eventOnHookedComponent() and eventOnClient());

identifying components within the component behavior model to be deployed to remote locations (see Part 2.2 of Kon; "a component configurator may be able to refer to components, ... running on different machines (remote locations) in a distributed system);

and for each component to be deployed to a remote location, identifying the remote location, and causing the component to be deployed to the remote location ("A distributed resource manager uses the specifications (deployment specifier) of the component hardware requirements to decide in which machine (identifying the remote location) the component (components within the component-behavior model) should be loaded (deployed)" (Kon, Part 3.2)).

As per claim 16, the rejection of claim 15 is incorporated, and further, Kon teaches **the method of claim 15, wherein identifying components to be deployed to remote locations involves examining a deployment specifier that indicates where each of the components within the component behavior model is to be deployed** ("A distributed resource manager uses the specifications (deployment specifier) of the

component hardware requirements to decide in which machine (where) the component (components within the component-behavior model) should be loaded (deployed)" (Kon Part 3.2)).

As per claims 31, and 32 they are computer-readable storage medium claims corresponding to method claim 15 and 16, respectively and are rejected under the same rationale.

As per claims 47, and 48 they are apparatus claims corresponding to method claim 15, and 16, respectively and are rejected under the same rationale.

Claim Rejections - 35 USC § 103

9. Claims 2, 18, and 34 rejected under 35 U.S.C. 103(a) as being unpatentable over Fowlow et al (U.S. 5,991,535) as applied to claims 1, 3-10, 12, 14, 17, 19-26, 28, 30, 33, 35-42, 44, 46 above, and further in view of Guthrie et al (U.S. 6,385,661).

As per claim 2, the rejection of claim 1 is incorporated, and further,

Fowlow teaches the use of encapsulation on distributed components of an application (column 6, lines 40-42).

Fowlow does not teach **automatically determining if any distributed components of the distributed application have not been encapsulated as local components; and for each distributed component that has not been encapsulated as a local component, automatically encapsulating the distributed component as**

a local component, so that the distributed component appears to be a local component.

Guthrie teaches an analogous method of enabling local components to implement remote components on a distributed object oriented system (column 3, lines 44-46, column 5, lines 23-29, column 1, lines 65-67) by **automatically determining** ("...generate at run time remote proxy classes as needed for inter-object communications ..." column 2 of Guthrie, lines 43-46) **if any distributed components of the distributed application have not been encapsulated as local components** (...determination is made regarding the need for a remote proxy class" refer to figure 2, item 30 of Guthrie, and column 5, lines 40-42, the need of a remote proxy class indicates a component has not been encapsulated as a local component) **and for each distributed component that has not been encapsulated as a local component, automatically encapsulating the distributed component as a local component, so that the distributed component appears to be a local component** ("if remote proxy class does not exist on client system, ...remote proxy class 23 is generated ..." refer to figure 2 of Guthrie, item 32 and column 5, lines 46-50, proxies are generated on local systems to facilitate access to objects on remote systems (column 3, lines 44-46)).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to use remote proxies disclosed by Guthrie to access remote or distributed objects disclosed by Fowlow.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to minimize compile and load time for the computer

program since remote proxy classes do not have to be generated at compile and load time (Guthrie column 4, lines 55-57).

As per claim 18 is a computer-readable storage medium claim corresponding to the method claim 2 and is rejected under the same rationale.

As per claim 31 is an apparatus claim corresponding to the method 2, and is rejected under rejected under the same rationale.

Claim Rejections - 35 USC § 103

10. Claims 11, 27, and 43 rejected under 35 U.S.C. 103(a) as being unpatentable over Fowlow et al (U.S. 5,991,535) as applied to claims 1, 3-10, 12, 14, 17, 19-26, 28, 30, 33, 35-42, 44, 46 as above, and further in view of Shah et al (U.S. 6,269,396).

As per claim 11, the rejection of claim 1 is incorporated, and further,

Fowlow does not specifically teach, **the method of claim 1, wherein causing the distributed component to be deployed to the remote location involves communicating with an application server at the remote location through a deployment server at the remote location, wherein the deployment server operates by: halting an application server process loading files for the distributed component onto the application server, setting preferences on the application**

server for the distributed component; and restarting the application server process.

Shah et al teaches in an analogous system in a distributed network, wherein steps of **deploying** (column 4, lines 66-67, column 5, line 1, and column 6, lines 47-49) **a distributed component** (configurable element set, column 5, line 2, column 6, lines 49-51) **to a remote location** (one or more nodes, column 5, line 1, and column 6, lines 33-35, lines 45-47) **involves communicating with an application server** (network figure 1, item 16) **at the remote location through a deployment server** (telecom platform, figure 1, items 14 and 15, telecom platform manager column 6, lines 45-46) **at the remote location, wherein the deployment server** (telecom platform, telecom platform manager) operates (manages coloumn5, lines 22-24) **by: halting an application server process loading files for the distributed component onto the application server** ("HALTED state" (column 5, lines 53)); **setting preferences on the application server for the distributed component** (configuration of nodes, column 7, lines 25-27, and lines 39-41); **and restarting the application server process** (a shutdown node (Halted) may recover automatically, column 5, lines 53-54).

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to use the invention disclosed by Fowlow to build the application using a visual composition builder, facilitating the development of object oriented applications and distributed object applications (Fowlow, column 2, lines 65-67, and column 3, lines 1-3), and, in the context of telecommunication, set up the network to distribute the application using a telecom platform manager disclosed by Shah.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to use the method and system disclosed by Shah to allow developers to write applications without having to master the intricacies of the underlying services, such as the operating system and the network, that perform the work on behalf of the application (Shah column 4, lines 13-17).

As per claim 27, is a computer-readable storage medium claim corresponding to method claim 11 and is rejected under the same rationale.

As per claim 43, is an apparatus claim corresponding to method claim 11 and is rejected under the same rationale.

Claim Rejections - 35 USC § 103

11. Claims 13, 29, and 45 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fowlow et al (U.S. 5,991,535) as applied to claims 1, 3-10, 12, 14, 17, 19-26, 28, 30, 33, 35-42, 44, 46 above, and further in view of Hammond (U.S. 6,637,020).

As per claim 13, the rejection of claim 1 is incorporated, and further

Fowlow teaches the use of encapsulation on developer objects in column 6, lines 40-42. And as specified in the rejection of claim 12, Fowlow teaches the use of COBRA objects as distributed components and that "an IDL data type may be encapsulated by

an Any object" (column 7, lines 64-65). An Any object refers to typecode of the encapsulated data, and generic encoding of the data (column 7, lines 64-67).

Fowlow does not teach **the method of claim 1, wherein a distributed programming component is an Enterprise JavaBean (EJB) that is encapsulated as a JavaBean by combining functionality of a home interface and a remote interface of the EJB into the JavaBean**.

Hammond teaches an analogous method for a distributed object oriented system, the step of **distributed programming component is an Enterprise JavaBean (EJB) that is encapsulated as a JavaBean** (JavaBean specification, column 4, line 47-48) **by combining functionality of a home interface and a remote interface of the EJB into the JavaBean** (see column 13, lines 37-67 of Hammond for an example of combining the functionality of a home interface (i.e. methods that contain the EJB lifecycle) and a remote interface (i.e. business methods exposed to clients)). He teaches the conversion of IDL COBRA files into JavaBean components.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use EJB components encapsulated as JavaBean components in the method recited by Fowlow in claim 1.

The modification would have been obvious because it would be advantageous for the user of any COBRA component object architecture to be able to create applications without relying on complete knowledge of the architecture and the labor intensive method of wiring the components together by hand by combining components dynamically into user applications (Hammond column 6, lines 34-38).

As per claim 29, is a computer-readable storage medium corresponding to method claim of claim 13 and rejected under the same rationale.

As per claim 45, is an apparatus claim corresponding to method claim 13 and rejected under the same rationale.

Conclusion

12. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Noreen Moiduddin whose telephone number is (703) 305-0538. The examiner can normally be reached on Monday, Tuesday, Wednesday, Thursday, and Friday 9:00 AM to 6:00 PM.

14. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703) 305-9662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

15. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should

you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100